Software performance of encryption algorithms and hash functions

Bart Preneel*

Katholieke Universiteit Leuven, ESAT-COSIC,
Kardinaal Mercierlaan 94, B–3001 Heverlee, Belgium
bart.preneel@esat.kuleuven.ac.be

Abstract

We examine the factors that influence the performance of cryptographic software. Two important elements are the high level design decisions and the optimal use of registers and cache memory. We also give performance numbers for a set of selected algorithms.

1 Introduction

An increasing number of software applications makes use of cryptographic algorithms. An important constraint is that the performance of the application should be influenced as little as possible by the introduction of cryptography. Two cryptographic primitives are required for high speed data processing, namely encryption algorithms and cryptographic hash functions. The first class is mainly used for confidentiality protection, while hash function are intended for information integrity. Because of its slow performance, asymmetric cryptography is not a viable alternative for symmetric algorithms for these applications.

Since the mid seventies the Data Encryption Standard (DES) has been a worldwide de facto standard, which was used for both confidentiality and authenticity; very little alternatives were available. During the last five years, an increasing number of symmetric algorithms has been proposed. One goal

*N.F.W.O. postdoctoral researcher, sponsored by the National Fund for Scientific Research (Belgium).

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

of these new algorithms is to take advantage of advances in general purpose processors to achieve a better performance in software and/or hardware. One should keep in mind that DES was designed taking into account hardware constraints of the mid seventies (the total size of the S-boxes is only 256 bytes). Moreover, researchers want to explore new alternatives based on open design criteria (the design criteria of DES were never published). An additional reason for the replacement of DES is its short key (56 bits), which makes it too vulnerable to exhaustive search [29]. This can however be solved easily by using triple DES, at the cost of a reduced performance. An important factor in the development of new algorithms is the recent progress in cryptanalysis (such as differential [3] and linear cryptanalysis [15]), which has brought new insights in the development of cryptographic algorithms and has made their design more of a science than an art. Furthermore, this has stimulated research on building blocks such as S-boxes.

The analysis of the performance of cryptographic algorithms is closely related to their security: high performance applications require an optimal trade-off between security and speed. In order to avoid the discussion of specific details of the large number of algorithms available, we emphasize the influence of high level design decisions and of the choice of the elementary building blocks on performance.

In this extended abstract, we will use the following examples of symmetric algorithms:

block ciphers: DES [8], FEAL [18], IDEA [13], Khufu [17], LOKI [5], Blowfish [28], 3-WAY [6], SAFER [14], CAST [1, 11], RC5 [24].

stream ciphers: RC4 (proprietary), SEAL [27].

hash functions: MDC-2 [4], Snefru [16], MD2 [12], MD4 [22], MD5 [23], SHA [9, 10], RIPEMD [20].

While this collection contains the most popular algorithms, a selection of newer algorithms has been included as well.

2 Performance

Optimizing the performance of software and hardware is quite different. Fast hardware relies on parallelism and pipelining, while for software the access to the memory is a key to fast performance: the designer will try to minimize access to slow memory, and to stay as much "on chip" (registers and cache memory). This aspect becomes more and more important as the access

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

time to the memory seems to decrease more slowly than the cycle time of the processors. This suggests that faster cryptographic primitives will make use of logic and arithmetic operations available on a standard processor and of relatively small S-boxes (a few Kbyte). The advantage of S-boxes is that they can yield a strong nonlinear relation between input and output. S-boxes with 8 input bits and 32 output bits seem to be particularly suited for the current 32-bit architectures. Other less important aspects which influence software performance are word size, byte ordering, and carries, which tends to be processor dependent. Recent computer architectures such as the Pentium will also allow for parallel execution, and one can anticipate that the importance of parallelism in software will increase.

The reader will probably be quite well aware that there exists no such thing as the "software performance" of a given algorithm, even if figures are given for a specific processor. The key is again the use of memory: some measurements are obtained while repeatedly processing the same block, which will have only a very distant relation to the performance in a practical application where data are to be read from disk. The behavior of the cache memory is an essential factor here. On the other hand, one wants to measure the performance of the algorithm rather than of the computer. Other factors which can be very important include:

- equivalent representations and the use of tables: this can yield significant speed-ups, mainly for algorithms which are not designed specifically for software. A widely used 'trick' is to rewrite the code for a different key, but this involves some practical problems and some security risks;

- quality of the compiler: for high level languages, good compilers (with the right optimizations activated) can produce code which is almost an order of magnitude faster; hand coded assembly language sometimes results in significant improvements by using processor instructions such as rotate which are not implemented in languages such as C, and by optimizing the use of registers.

One can ask the question whether one should try to optimize the design towards a single processor: designing and reviewing a cryptographic algorithm will take several years, and by that time the processor will probably be outdated. But, most processors are downward compatible, and if one tunes an algorithm to a recent processor without exploiting particular features (such as the availability of certain carries), it is very likely to achieve

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

a high speed on most other processors as well. Finally it should be noted that the evolution of processors on smart cards is significantly slower than that of general purpose processors.

3 Block Ciphers and Stream Ciphers

For block ciphers, the most important high level design choice is between Feistel ciphers (DES, FEAL, Khufu, LOKI, Blowfish, CAST) and substitution-permutation networks (SAFER, 3-WAY, and RC5). IDEA uses a generalization of the Feistel concept. The advantage of the substitution-permutation networks is that every bit of the intermediate ciphertext is modified in every round. Apparently this makes it more difficult to find high probability differential characteristics and linear approximations. On the other hand every operation has to be invertible, and the encryption and decryption operation are in general different. This is not a disadvantage in software, but poses some problem for hardware implementations (note that in 3-WAY, which is hardware oriented, both operations are identical). For Feistel ciphers, fewer constraints are imposed on the round function and more cryptanalytic experience is currently available.

A crucial component in every algorithm is the *nonlinear* element, which usually consists of S-boxes. RC5 is an exception, since it uses data dependent rotations. 3-WAY uses one small S-box (with 3 input bits and 3 output bits, denoted with $(3 \rightarrow 3)$. For the other algorithms, we have the following numbers: DES $(6 \rightarrow 4)$, FEAL and SAFER $(8 \rightarrow 8)$, LOKI $(12 \rightarrow 8)$, Khufu, Blowfish, and CAST $(8 \rightarrow 32)$, and IDEA $(16 \rightarrow 16)$. The S-box of FEAL is a modular addition, which is a quite weak operation, while both SAFER and LOKI use exponentiation, CAST uses bent components, and Khufu and Blowfish use secret values. IDEA relies on three incompatible arithmetic and logic operations.

The *diffusion* of information is usually performed with linear operations. DES, LOKI, and 3-WAY use a bit level permutation. FEAL and IDEA use a structure with different consecutive S-boxes, while both Blowfish and CAST add the output of 4 S-boxes. SAFER uses an operation with a butterfly structure, denoted as the pseudo-Hadamard transform.

Recent cryptanalytic results show that using a too simple *key schedule* is probably not a good idea, since it might introduce new weaknesses. On the other hand, a complex key schedule will be a problem if the key is changed often (this happens for example after every encryption in most

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

hash functions based on block ciphers). There is also a point of diminishing returns since an attacker can always try to find the round keys rather than the key itself. Key dependent S-boxes (such as in Blowfish and Khufu) do not foster a quick key change.

From the cryptanalytic results available, we can draw the conclusion that resistance against linear or differential techniques is not a property of S-boxes or of linear mappings, but rather of the combination of both. However, for a similar diffusion structure, increasing the size of the S-boxes will increase this resistance (e.g., LOKI versus DES).

Most stream ciphers in the literature are based on shift registers which operate at bit level. It turns out that they are in general not too fast in software (e.g., the shrinking generator). Recently, several faster software oriented proposals have been made (see for example [2, 19]). Two important examples are RC4, which is a widely used proprietary algorithm, and SEAL, which is extremely fast.

4 Hash Functions

Almost all practical collision resistant hash functions are iterated hash functions, which means that they compress their input block by block by applying a compression function with fixed size inputs:

$$H_0 = IV; \qquad H_i = f(H_{i-1}, X_i), 1 \leq i \leq t \qquad h(x) = H_t\,.$$

Here X_i denoted the ith message block, while H_i denotes the chaining variable. The main design decision is whether the compression function f itself should be collision resistant: this decision affects the complete design of the function, since a collision resistant compression function should treat both its arguments the same way.

The main argument in favor of a collision resistant and one-way compression function is the provable reduction of the security of the hash function to that of the compression function. Another advantage of this construction is that the choice of a specific IV is not very important: the scheme should be secure with any IV. Also, the size of the chaining variables can be increased at the cost of a decreased performance; this corresponds to a trade-off between security and speed. Moreover, the round function can be used directly in applications where the size of the input is fixed. Finally hashing can be done in a parallel for a variant based on a tree construction.

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

On the other hand, from analyzing the hashing process, it follows that the roles of H_i and X_i are essentially different. This means that it seems natural to impose different conditions on both inputs of the round function. One can hope that loosening the requirements will improve the performance of the hash function.

Hash functions which consist of a compression function with only a single input include MD2 and Snefru; designs that treat chaining variables and message blocks differently include MD4 and its variants. On the other hand, the only round functions for which no collisions were found are those of Snefru with more than 6 passes, MD4, SHA, and RIPE-MD (MD5 is excluded). This is rather surprising, as the MD4 variants are designs of the second type.

5 Indirect Constructions

As mentioned in the introduction, the popularity of DES has lead to construction of hash functions based on block ciphers. Although it not yet possible to formalize the properties of a block cipher that are required to make it suited for such an application, significant progress has been made on the development of practical constructions (an important example is MDC-2). However, these schemes are in general slow, since at least two encryptions are required to hash a single block, and almost all constructions modify the key after each encryption.

Alternatively, the availability of fast hash functions has prompted designs of block ciphers and stream ciphers based on hash functions. While these schemes are fast, caution is required since the use in these constructions imposes additional conditions on the hash function, which they may or may not satisfy.

6 Software performance

Table 1 gives an overview of the speed of the most important algorithms. The timings were performed on a 16 MHz IBM PS/2 Model 80 with a 80386 processor, on a PC with a 60 MHz 80586 (Pentium) processor, and on a DEC 3000/400 with a 133 MHz Alpha processor. This gives an idea of the evolution in software performance over the last seven years. The PC implementations were written by A. Bosselaers. Most of them use additional memory to improve the speed. The C-code was compiled with a 32-bit compiler in protected mode. For several algorithms an implementation in

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

assembly language was written as well. The C-code for the Alpha was written by M. Roe [25, 26].

For comparison, we have included the timings for a modular squaring and exponentiation with a short exponent. In this case a 1024-bit modulus was chosen, and no use was made of the Chinese remainder theorem to speed up the computations. For the exponentiation the fourth Fermat number $(2^{16} + 1)$ was used as exponent.

The table is far from complete, but it gives a good idea of the differences between the algorithms and between the machines. Note that in some cases the C-compiler produces code that is almost optimal. By comparing the different entries, one can conclude that the DES implementation on the PC is better optimized than that on the Alpha. Moreover, highly optimized DES code is not much slower than most 'fast' block ciphers, but it requires a 64 Kbyte table, which is a problem for some environments. From the DES figures it can be derived that MDC-2 will run at about 100 Kbit/s on the 80386. Some algorithms like Snefru and SHA perform relatively better on a RISC processor (such as the Alpha), where the complete internal state can be stored in the registers. Also remarkable is the fast performance of SEAL, that was tuned by its designers to be fast on 80x86 machines (use of registers and operation on registers).

References

[1] C.M. Adams, "Simple and effective key scheduling for symmetric ciphers," *Proc. of SAC'94, Workshop on Selected Areas in Cryptography.*

[2] R. Anderson, Ed. *Fast Software Encryption, LNCS 809*, Springer-Verlag, 1994.

[3] E. Biham and A. Shamir, *"Differential Cryptanalysis of the Data Encryption Standard,"* Springer-Verlag, 1993.

[4] B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel, and M. Schilling, *"Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function,"* U.S. Patent Number 4,908,861, March 13, 1990.

[5] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry, " Improving resistance to differential cryptanalysis and the redesign of LOKI," *Advances in Cryptology, Proc. Asiacrypt'91, LNCS 739*, H. Imai, R.L. Rivest, and T. Matsumoto, Eds., Springer-Verlag, 1993, pp. 36–50.

[6] J. Daemen, R. Govaerts, and J. Vandewalle, "A new approach to block cipher design," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 18–32.

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

type	algorithm	16 MHz 80386		60 MHz 80586		133 MHz
		C	Ass.	C	Ass.	Alpha C
block	DES†	0.13	0.20			
cipher	DES	0.51	0.66	7.0	7.7	1.86
	IDEA		0.52			
	FEAL-16	0.33	0.62			
	Blowfish-16					11.6
	3-WAY					5.2
	RC5‡					7.7
	SAFER∗					7.68
stream	RC4					15.4
cipher	SEAL				80.0	115.0
hash	MD2	0.078	0.078	1.7	1.7	0.76
function	MD4	2.67	6.27	54.1	72.5	78.8
	MD5	1.85	4.40	37.2	51.3	60.0
	SHA	0.71	1.37	14.7	20.9	
	SHA-rev.			13.7	19.5	41.5
	RIPEMD	1.33	3.10	26.8	35.7	48.0
	Snefru-8	0.27	0.27			
modular	squaring	0.0185	0.0973	0.267		
arithmetic	exponent.	0.00144	0.0071	0.00204		

† includes key scheduling.

‡ RC5-32/32/20 (32-bit words, 32 rounds, 20-byte key).

∗ SAFER-K64 with 6 rounds.

Table 1: Performance in Mbit/s of several block ciphers, stream ciphers, and hash functions on a 16 MHz 80386, a 60 Mhz 80586, and a 133 MHz Alpha.

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

[7] I.B. Damgård, "A design principle for hash functions," *Advances in Cryptology, Proc. Crypto'89, LNCS 435*, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416–427.

[8] *"Data Encryption Standard,"* Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[9] *"Secure Hash Standard,"* Federal Information Processing Standard (FIPS), Publication 180, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., 1993.

[10] FIPS 180, *"Proposed Revision of FIPS 180, Secure Hash Standard,"* Federal Register, July 11, 1994.

[11] H.M. Heys and S.E. Tavares, "On the security of the CAST encryption algorithm," *Canadian Conference on Electrical and Computer Engineering*, Sept. 1994, Halifax, Canada.

[12] B.S. Kaliski, "The MD2 Message-Digest algorithm," *Request for Comments (RFC) 1319*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[13] X. Lai, J.L. Massey and S. Murphy, "Markov ciphers and differential cryptanalysis," *Advances in Cryptology, Proc. Eurocrypt'91, LNCS 547*, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 17–38.

[14] J.L. Massey, "SAFER-K64: A byte oriented block-ciphering algorithm," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 1–17.

[15] M. Matsui, "Linear cryptanalysis method for DES cipher," *Advances in Cryptology, Proc. Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386–397.

[16] R. Merkle, "Fast software encryption functions," *Advances in Cryptology, Proc. Crypto'91, LNCS 576*, J. Feigenbaum, Ed., Springer-Verlag, 1992, pp. 476–501.

[17] R. Merkle, "A fast software one-way hash function," *Journal of Cryptology*, Vol. 3, No. 1, 1990, pp. 43–58.

[18] S. Miyaguchi, "The FEAL cipher family," *Advances in Cryptology, Proc. Crypto'90, LNCS 537*, S. Vanstone, Ed., Springer-Verlag, 1991, pp. 627–638.

[19] B. Preneel, Ed. *Fast Software Encryption, LNCS*, Springer-Verlag, to appear.

[20] "Race Integrity Primitives Evaluation (RIPE): final report," RACE 1040, 1993.

[21] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications ACM*, Vol. 21, February 1978, pp. 120–126.

[22] R.L. Rivest, "The MD4 message-digest algorithm," *Request for Comments (RFC) 1320*, Internet Activities Board, Internet Privacy Task Force, April 1992.

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.

[23] R.L. Rivest, "The MD5 message-digest algorithm," *Request for Comments (RFC) 1321*, Internet Activities Board, Internet Privacy Task Force, April 1992.

[24] R.L. Rivest, "The RC5 encryption algorithm," *Fast Software Encryption, LNCS*, B. Preneel, Ed., Springer-Verlag, to appear.

[25] M. Roe, "Performance of symmetric ciphers and one-way hash functions," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 83–89.

[26] M. Roe, "Performance of block ciphers and hash functions — one year later," *Fast Software Encryption, LNCS*, B. Preneel, Ed., Springer-Verlag, to appear.

[27] Ph. Rogaway and D. Coppersmith, "A software-optimised encryption algorithm," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 56–63.

[28] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," *Fast Software Encryption, LNCS 809*, R. Anderson, Ed., Springer-Verlag, 1994, pp. 191–204.

[29] M.J. Wiener, "Efficient DES key search," *Technical Report TR-244*, School of Computer Science, Carleton University, Ottawa, Canada, May 1994. Presented at the rump session of Crypto'93.

Appeared in *Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995*, pp. 89–98, 1995.